MIC 2005

Search-based Testing

Dr. Joachim Wegener

DaimlerChrysler AG, Research and Technology, Berlin
Software Analysis and Testing

Prof. Mark Harman

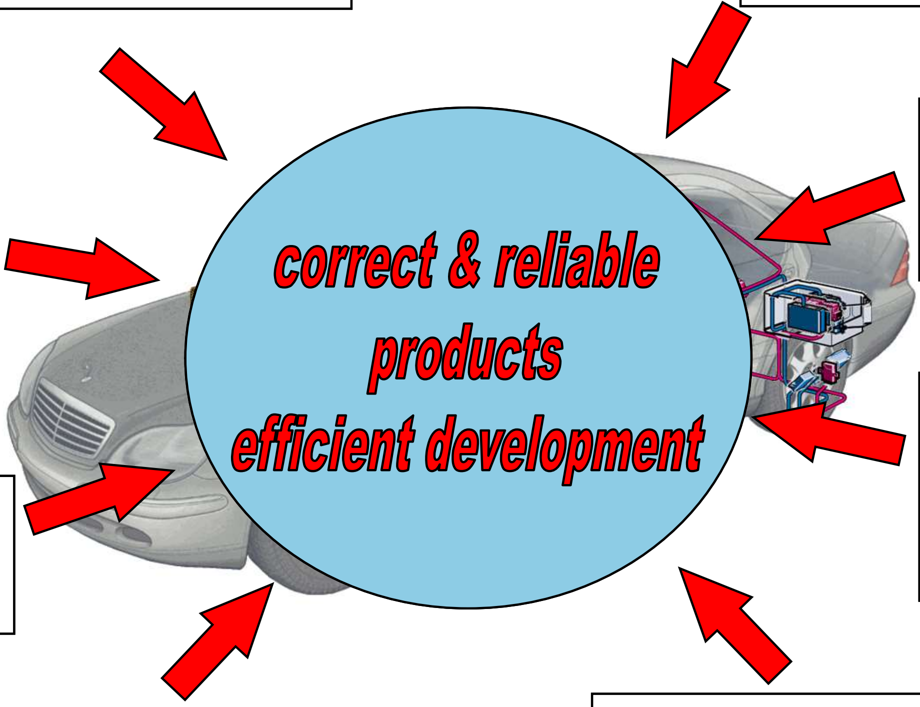
King’s College, London
Department of Computer Science

Outline

- Motivation
- Introduction to Search-based Testing
- Representation
- Fitness Functions
- Landscapes
- Managing Complexity
- Challenges and Conclusion

Industrial Viewpoint

- Increasing amount of software in products in almost all areas
- Software as differentiating factor for competitive advantage
- Cost-effective development and time-to-market crucial for competitiveness
- Increasing complexity of systems



correct & reliable products
efficient development

- High consequential costs due to faults
- Increasing significance of software in safety relevant areas
- High costs and greater time commitment for system development and maintenance
- High requirements for software based systems due to norms, standards and legal regulations

Academic Viewpoint

Publishing

Grant Funding

Cost of gowns for graduation

Time to get to mass

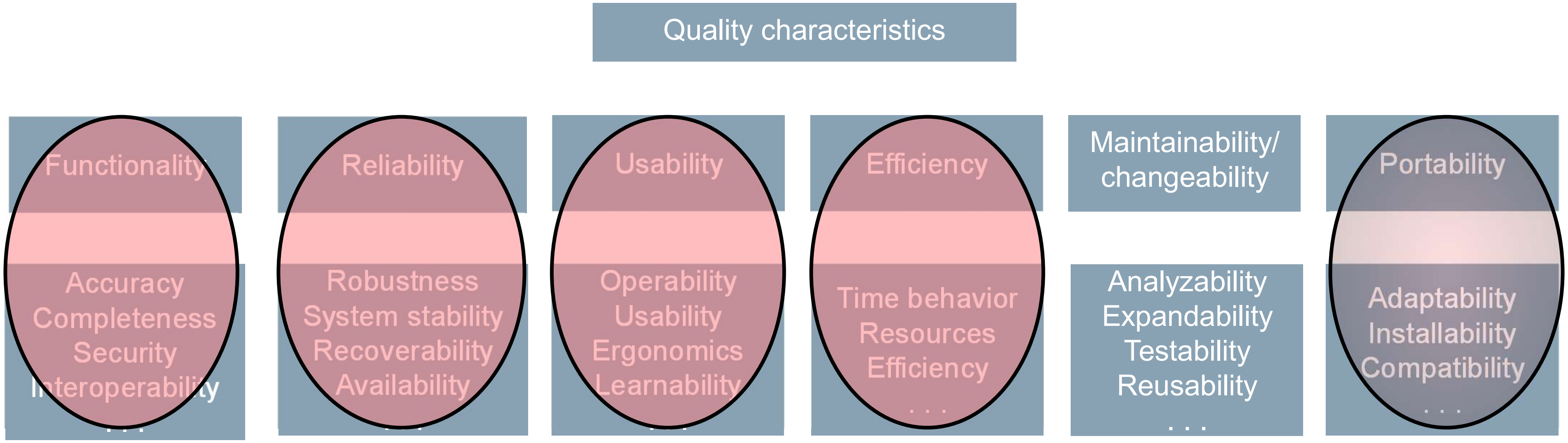
Sufficient coffee to make the next cup

Testing is Everywhere

System execution with selected test data aiming at

- detecting errors in the system under test and
- gaining confidence in the correct functioning of the system under test

Quality criteria (ISO 9126)


Quality characteristics

Functionality: Accuracy, Completeness, Security, Interoperability, . . .

Reliability: Robustness, System stability, Recoverability, Availability, . . .

Usability: Operability, Usability, Ergonomics, Learnability, . . .

Efficiency: Time behavior, Resources, Efficiency, . . .

Maintainability/ changeability: Analyzability, Expandability, Testability, Reusability, . . .

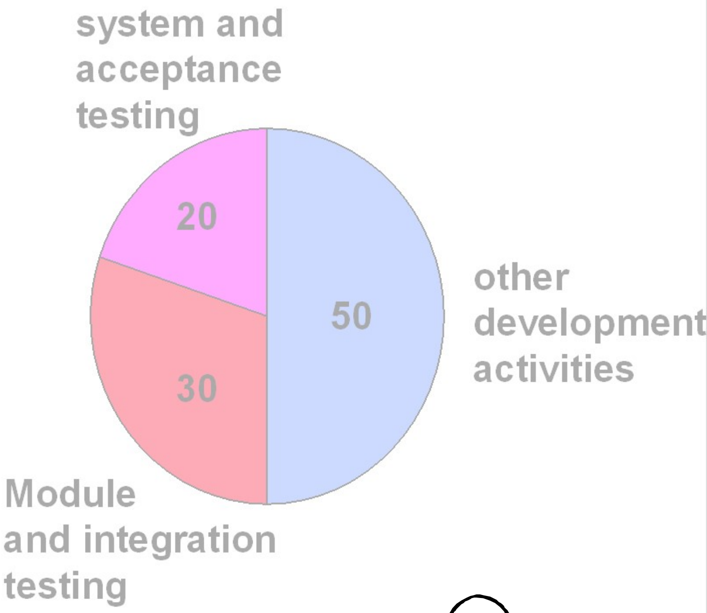
Portability: Adaptability, Installability, Compatibility, . . .

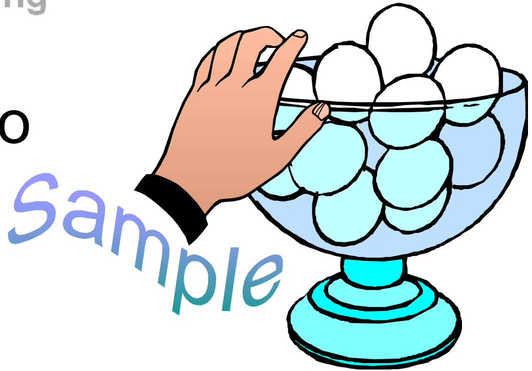
Characteristics to be addressed by Testing

Testing is Important and Expensive

Testing is the most important analytical quality assurance method

- ⊕ takes into consideration the real application environment (e.g. target
- ⊕ computer, compiler)
- ● tests dynamic system behavior (e.g. run-time behavior, memory space requirements)
- ⊖ testing carries a considerable cost-factor within system development
- ⊖ exhaustive test is usually impossible
  - → test data have to be selected according to certain test criteria (test methods)
  - → usually performed manually

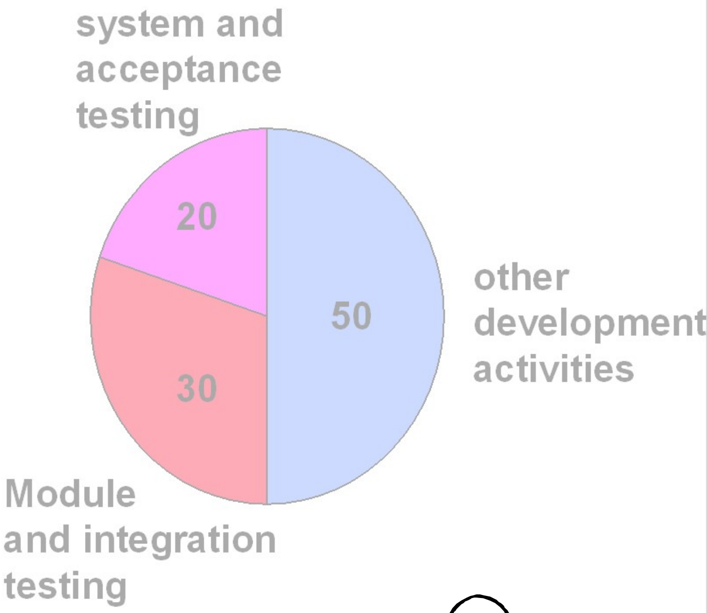




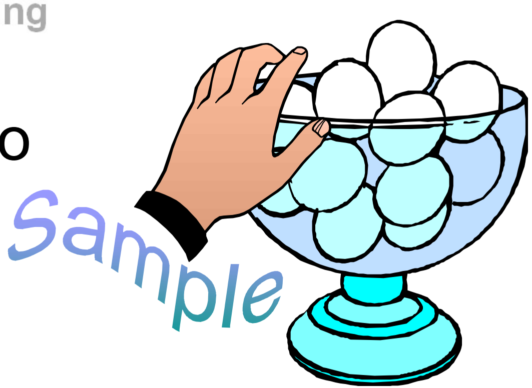
Testing is Important and Expensive

Testing is the most important analytical quality assurance method

- ⊕ takes into consideration the real application environment (e.g. target
- ⊕ computer, compiler)
- ● tests dynamic system behavior (e.g. run-time behavior, memory space requirements)
- ⊖ testing carries a considerable cost-factor within system development



- ⊖ exhaustive test is usually impossible
  - test data have to be selected according to certain test criteria (test methods)
  - usually performed ***manually***

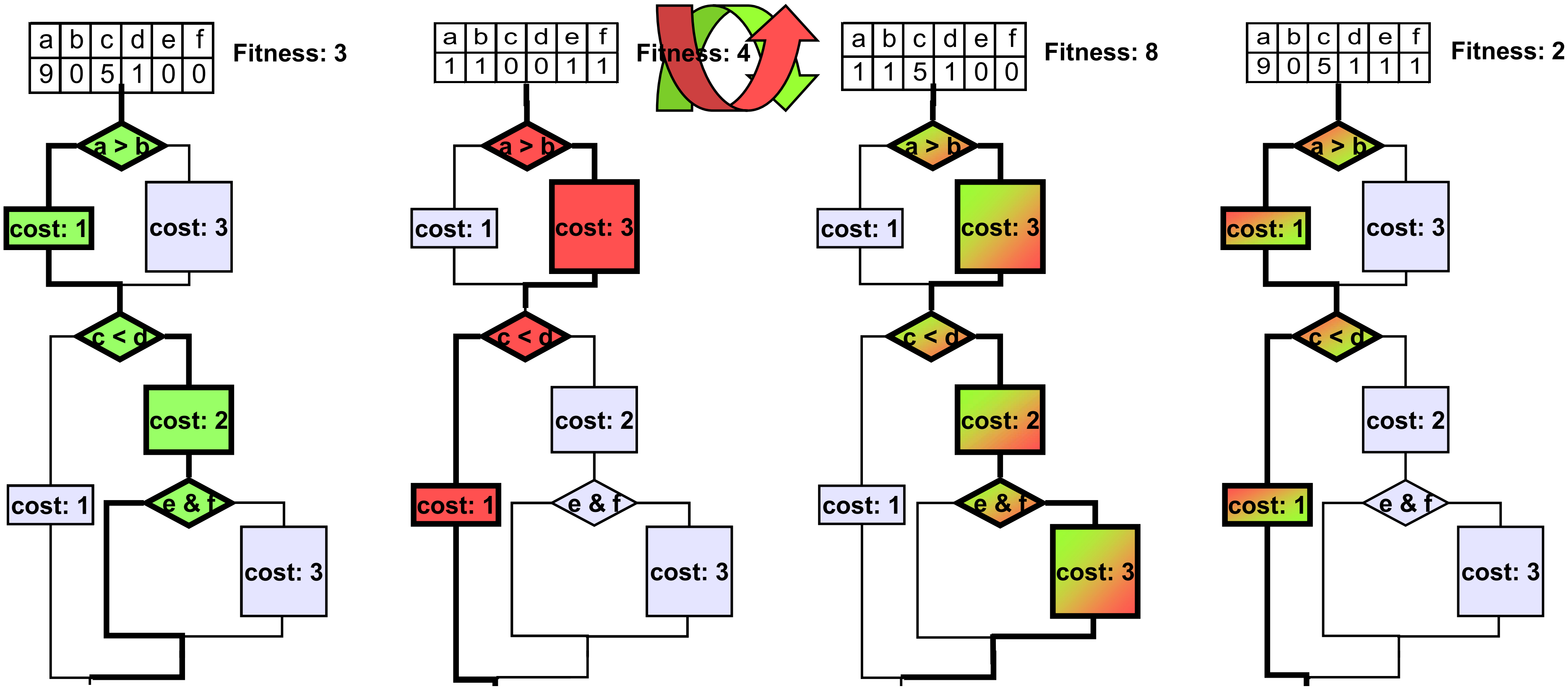


Introduction to Search-Based Testing

- Human (manual) search for faults
  - Expensive
  - Error prone
- We aim to fully automate the search
  - to improve test quality and
  - to increase test efficiency
  - ➢ Representation of input
    - Easy and obvious
  - ➢ Fitness function
    - Capture the aspect we are trying to test

Individual Representations for Testing

- input parameters of system under test form the variables to be generated by EC
- individuals could be used 1:1 as input parameters



a	b	c	d	e	f
9	0	5	1	0	0

Fitness: 3

a	b	c	d	e	f
1	1	0	0	1	1

Fitness: 4

a	b	c	d	e	f
1	1	5	1	0	0

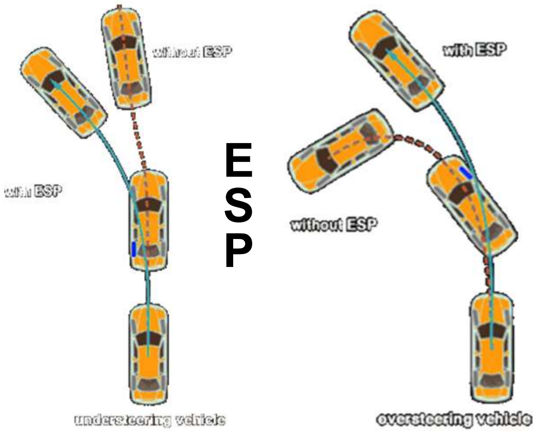
Fitness: 8

a	b	c	d	e	f
9	0	5	1	1	1

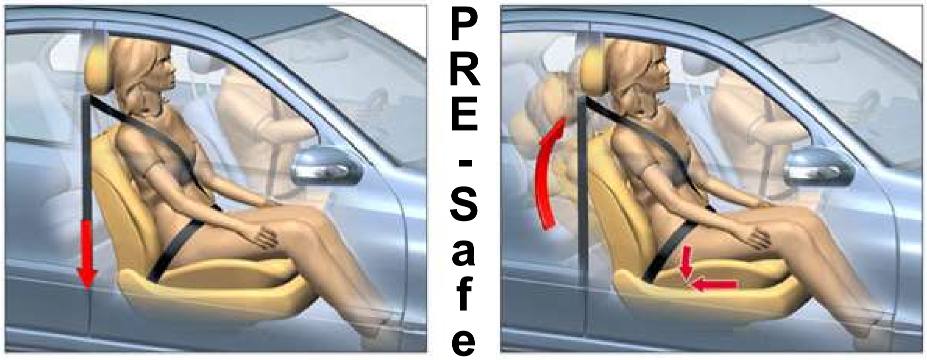
Fitness: 2

Individual Representations for Testing

- Often complex transformations of individuals are necessary to gain test data, e.g. if
  - Internal states got involved
  - Curve traces have to be generated



ESP

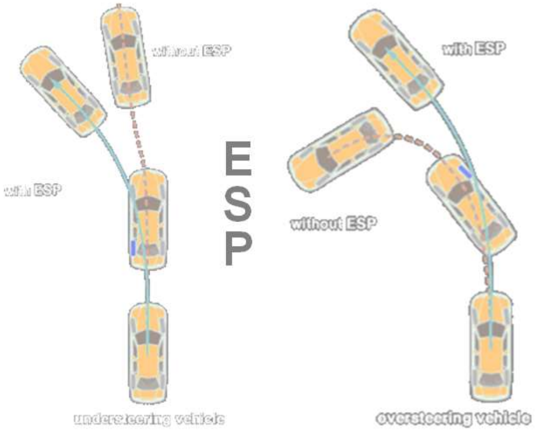


PRE-Safe

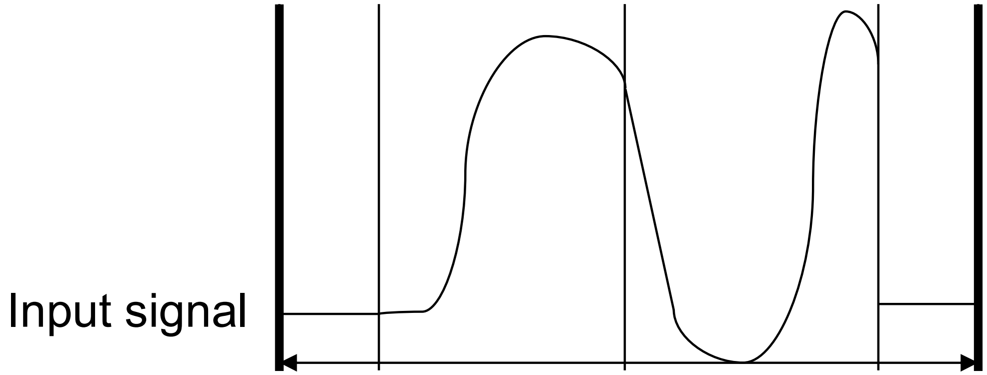
Distronic

Individual Representations for Testing

- Often complex transformations of individuals are necessary to gain test data, e.g. if
  - Internal states got involved
  - Curve traces have to be generated

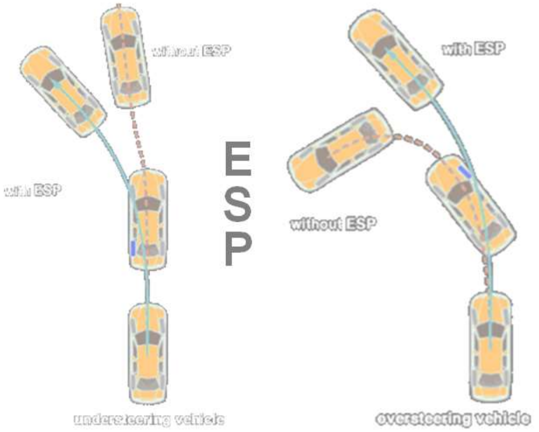


- Length of input signal

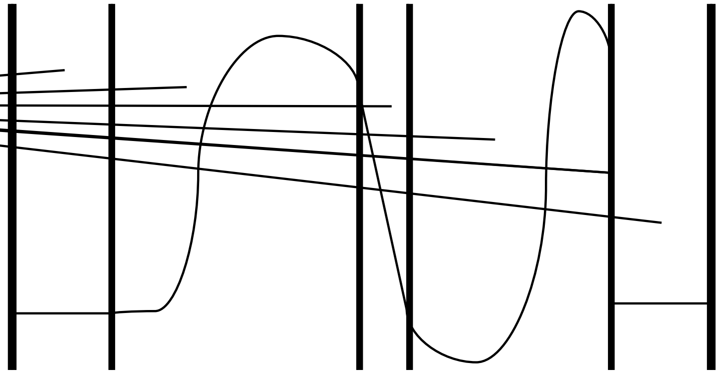


Individual Representations for Testing

- Often complex transformations of individuals are necessary to gain test data, e.g. if
  - Internal states got involved
  - Curve traces have to be generated

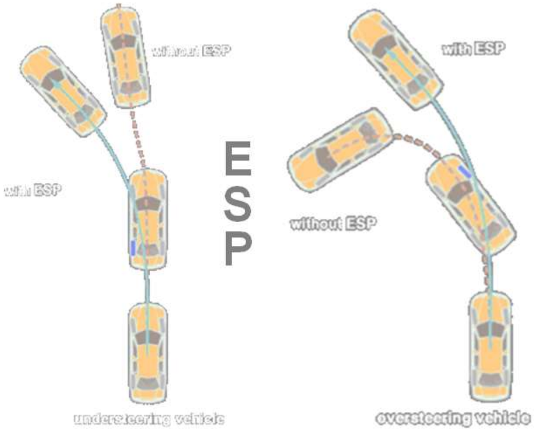


- Length of input signal
- Number of signal sections

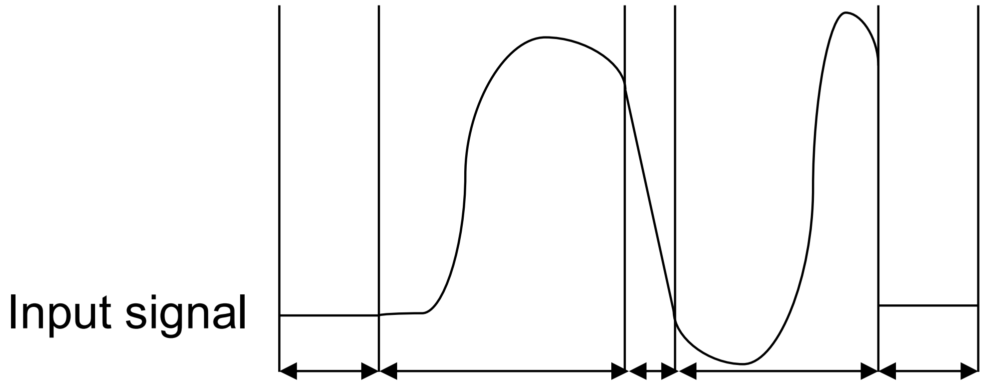


Individual Representations for Testing

- Often complex transformations of individuals are necessary to gain test data, e.g. if
  - Internal states got involved
  - Curve traces have to be generated

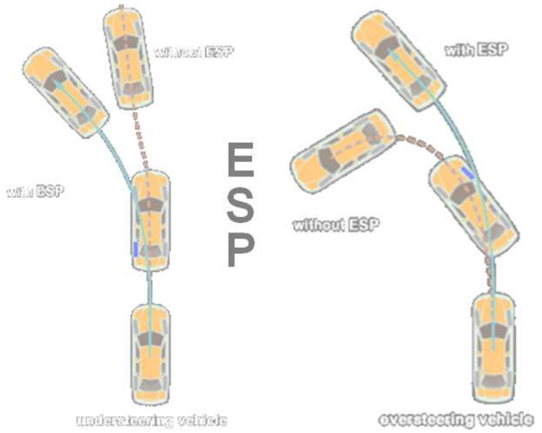


  - Length of input signal
  - Number of signal sections
  - Length of signal sections

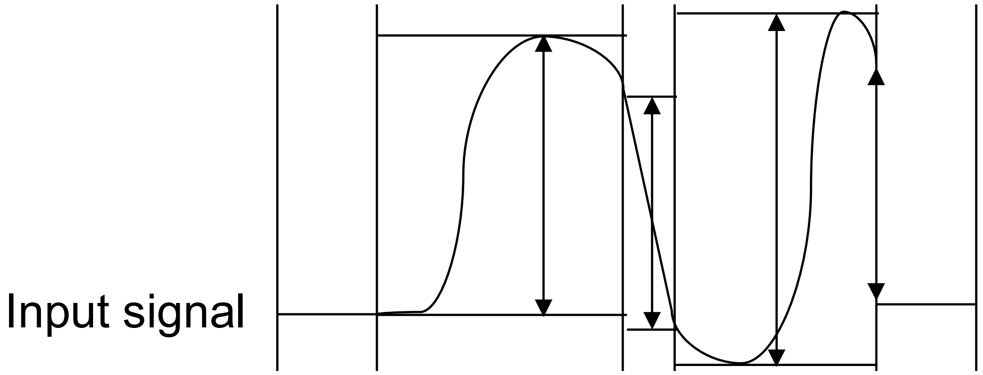


Individual Representations for Testing

- Often complex transformations of individuals are necessary to gain test data, e.g. if
  - Internal states got involved
  - Curve traces have to be generated

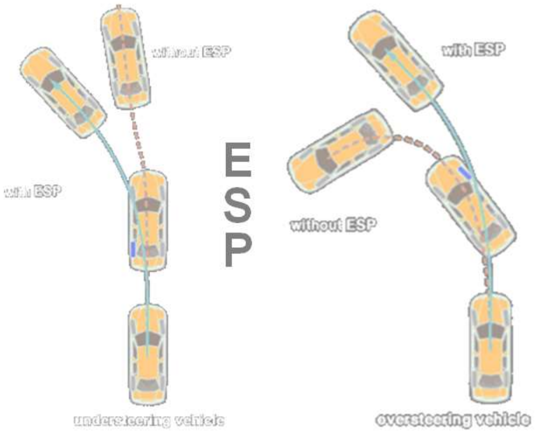


- Length of input signal
- Number of signal sections
- Length of signal sections
- Amplitude of signal sections

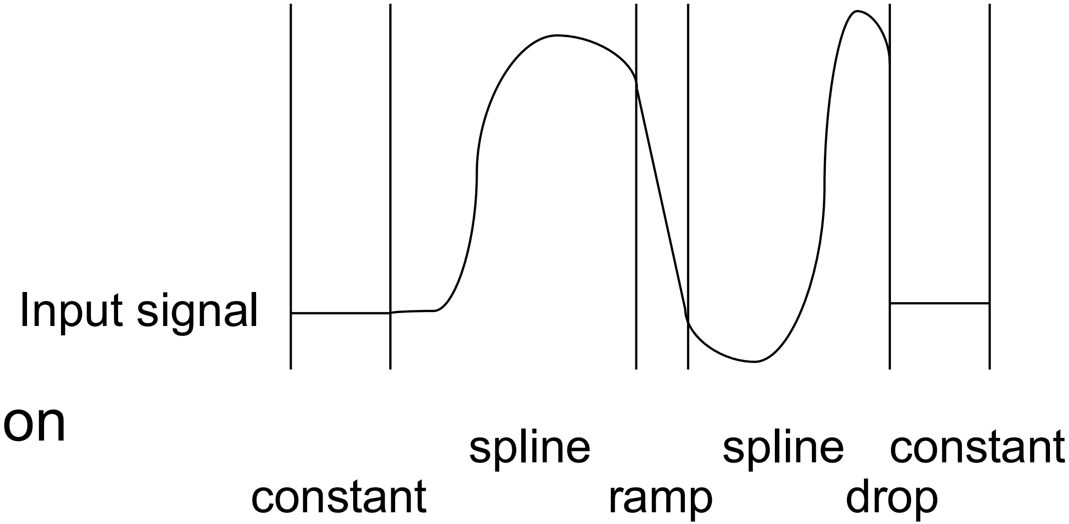


Individual Representations for Testing

- Often complex transformations of individuals are necessary to gain test data, e.g. if
  - Internal states got involved
  - Curve traces have to be generated



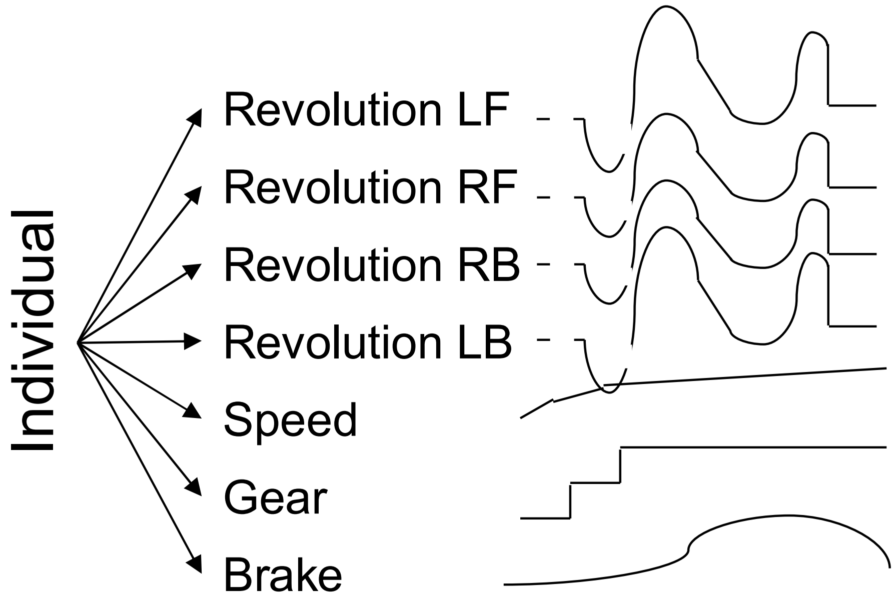
- Length of input signal
- Number of signal sections
- Length of signal sections
- Amplitude of signal sections
- Function type for signal section



Individual Representations for Testing

- Often complex transformations of individuals are necessary to gain test data, e.g. if
  - Internal states got involved
  - Curve traces have to be generated

- Length of input signal
- Number of signal sections
- Length of signal sections
- Amplitude of signal sections
- Function type for signal section

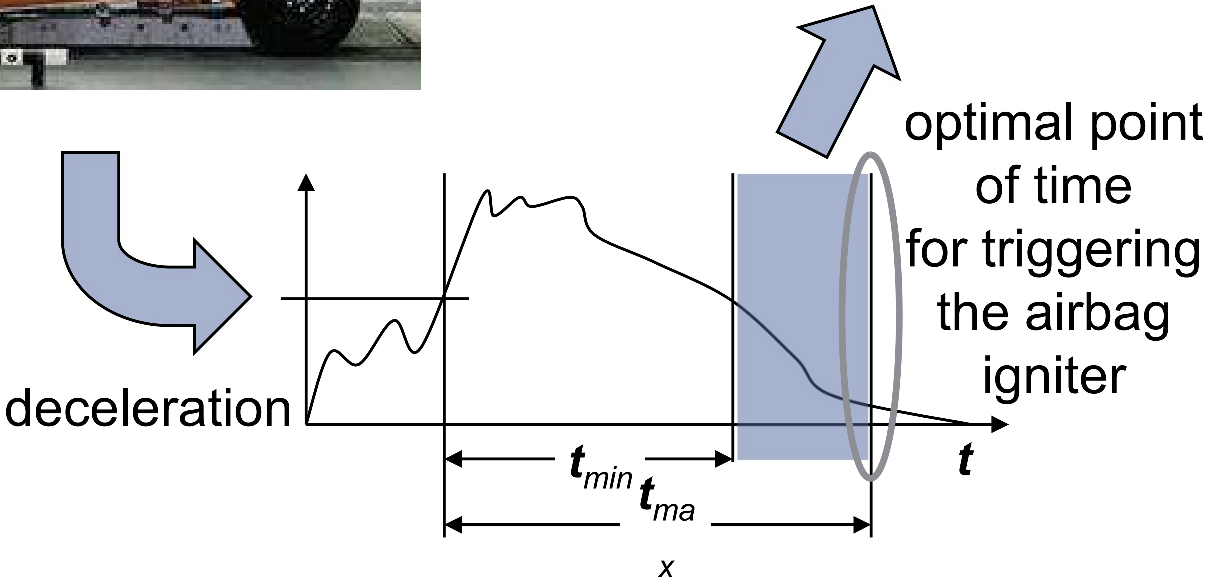


Overall process

- The search aims to find a single test datum
- The process is repeated to find a set of test data
- A single search may find others as a side product

Real-Time Testing

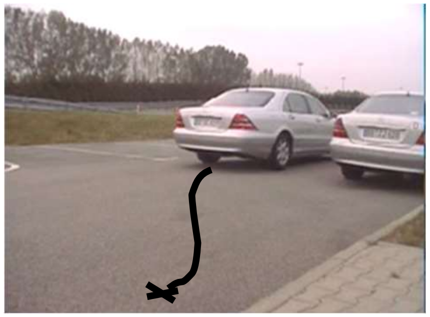
- Most embedded systems have to fulfil timing constraints (technical processes to be controlled, operational comfort)



Autonomous Parking System

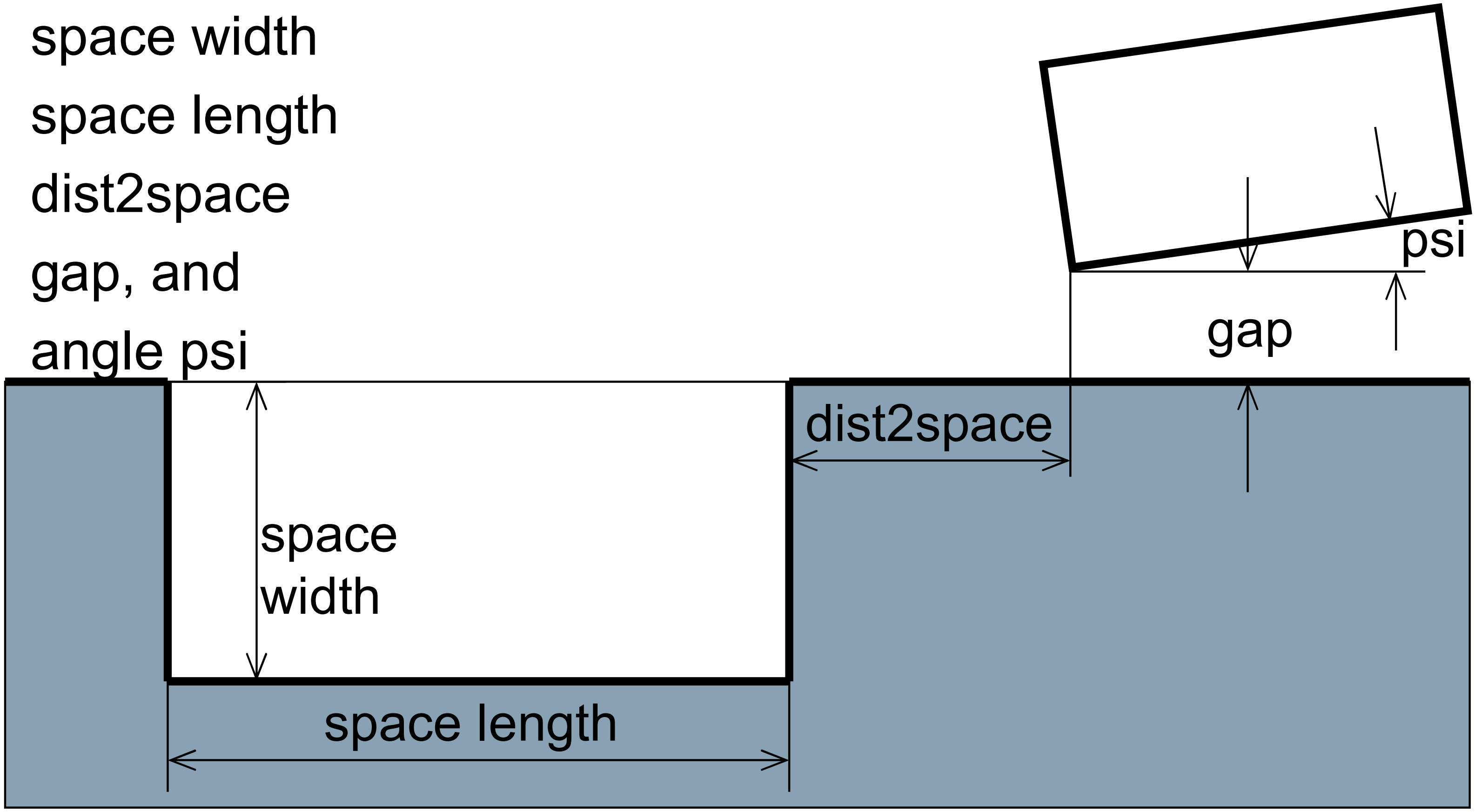
Steps:

- Measuring the size of the parking space using environmental sensors and parking space model
- Signaling sufficient sized parking spaces to the driver
- If parking is committed by the driver:
  - Determine the position of the car with respect to the parking space
  - Plan the trajectory path for the parking maneuver
- Drive the car into the parking space autonomously



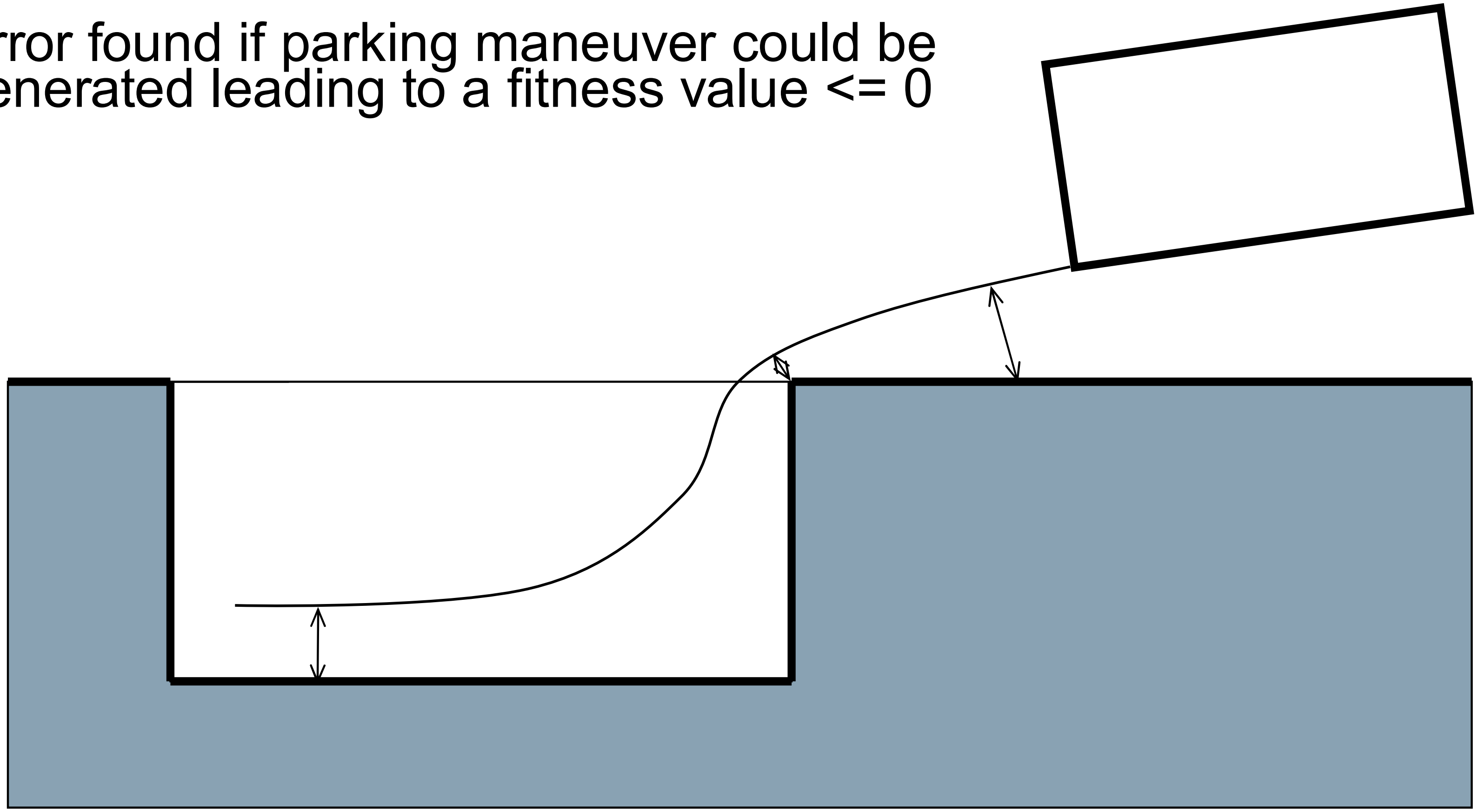
Autonomous Parking System

- Generation of parking scenarios by evolutionary algorithms varying
  - space width
  - space length
  - dist2space
  - gap, and
  - angle psi



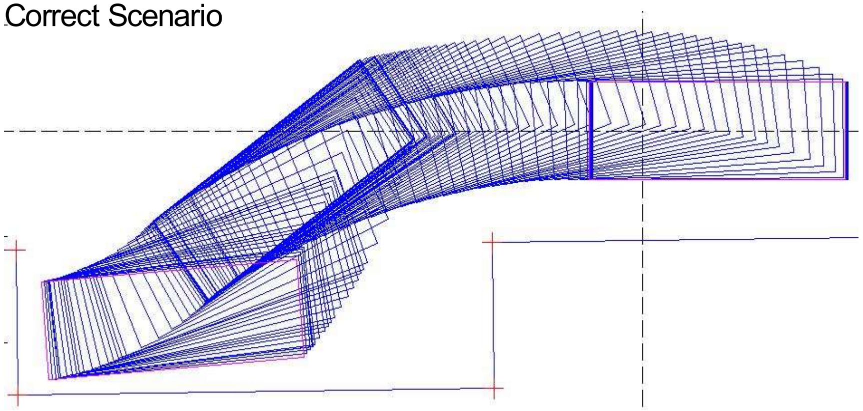
Autonomous Parking System

- Selection of smallest distance between car and collision area as fitness value (negative values also allowed)
- Error found if parking maneuver could be generated leading to a fitness value <= 0



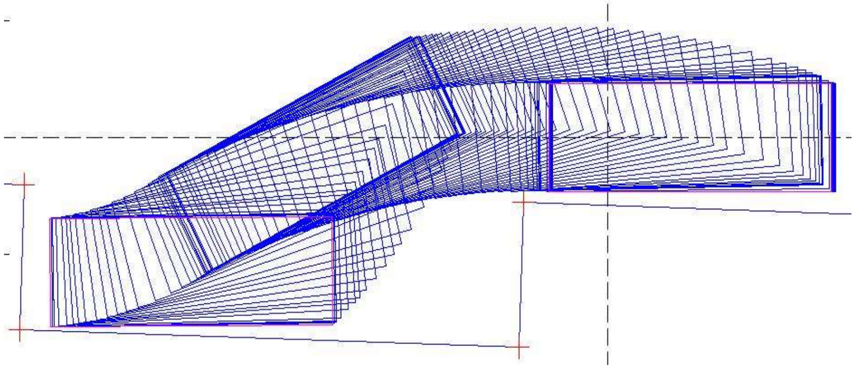
Autonomous Parking System

Correct Scenario



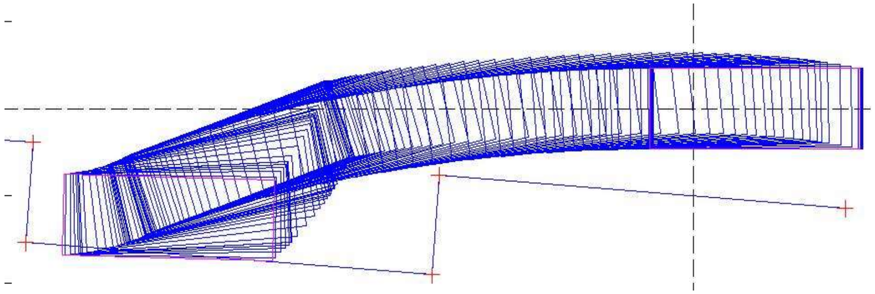
Generation 01 / Individual 13

Critical Scenario



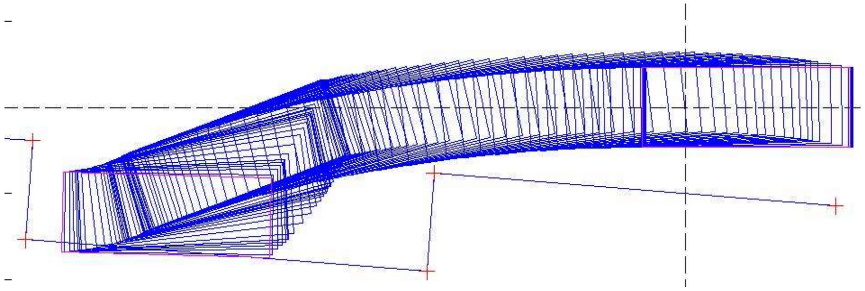
Generation 10 / Individual 02

Scenario leading to erroneous system behavior (edge entered collision area)



Generation 20 / Individual 06

Scenario leading to erroneous behavior (end-position in collision area)



Generation 20 / Individual 05

Fitness Functions

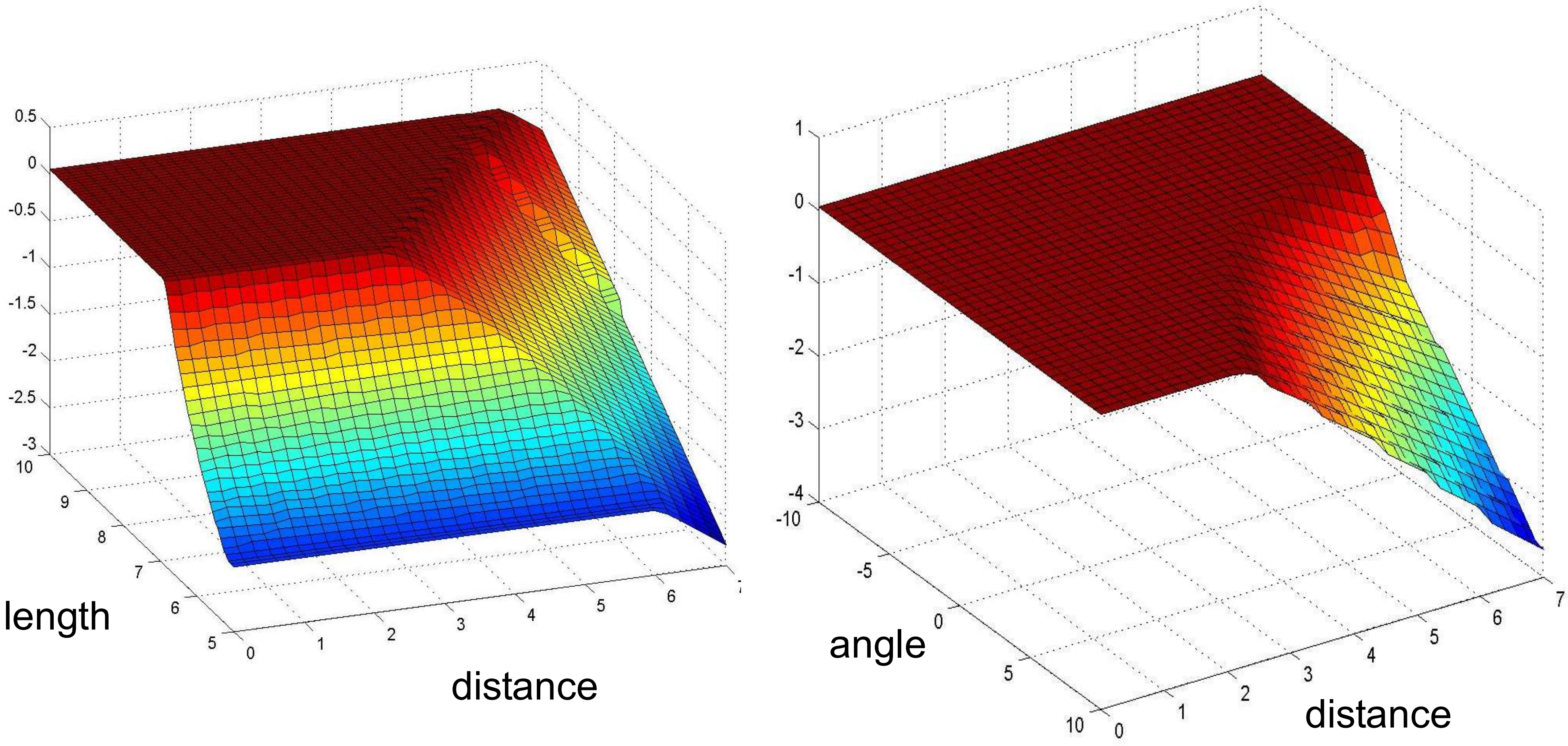
- Fitness functions are required for each kind of testing

Temporal	Search for extreme execution times
Functional	Search for logical error
Safety	Search for constraint violation
Structural path	Search for data which exercises desired

Variety of Landscapes

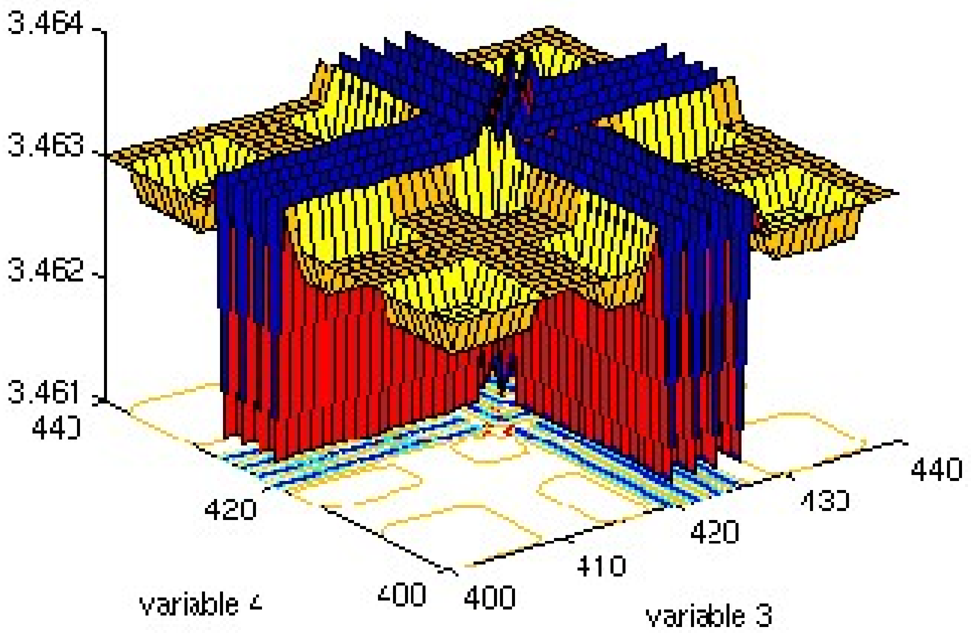
- There are many test scenarios
- There are many systems under test
- Each has its own landscape
- The features of the landscapes contain all the features that make the application of metaheuristics interesting

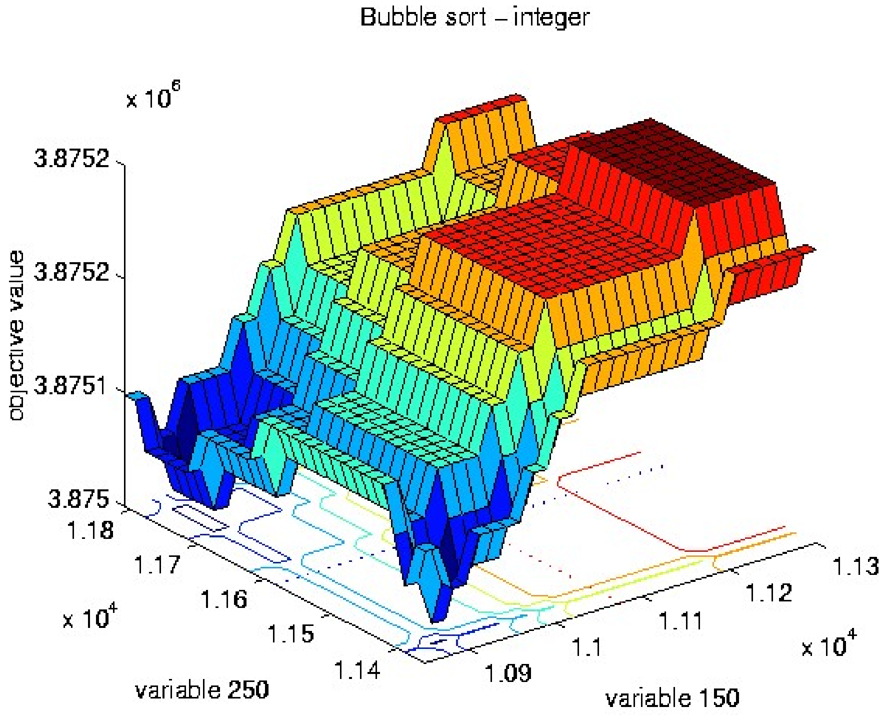
Parking System Testing



Relatively nice landscapes

Temporal Testing





Relatively nasty landscapes

Landscapes

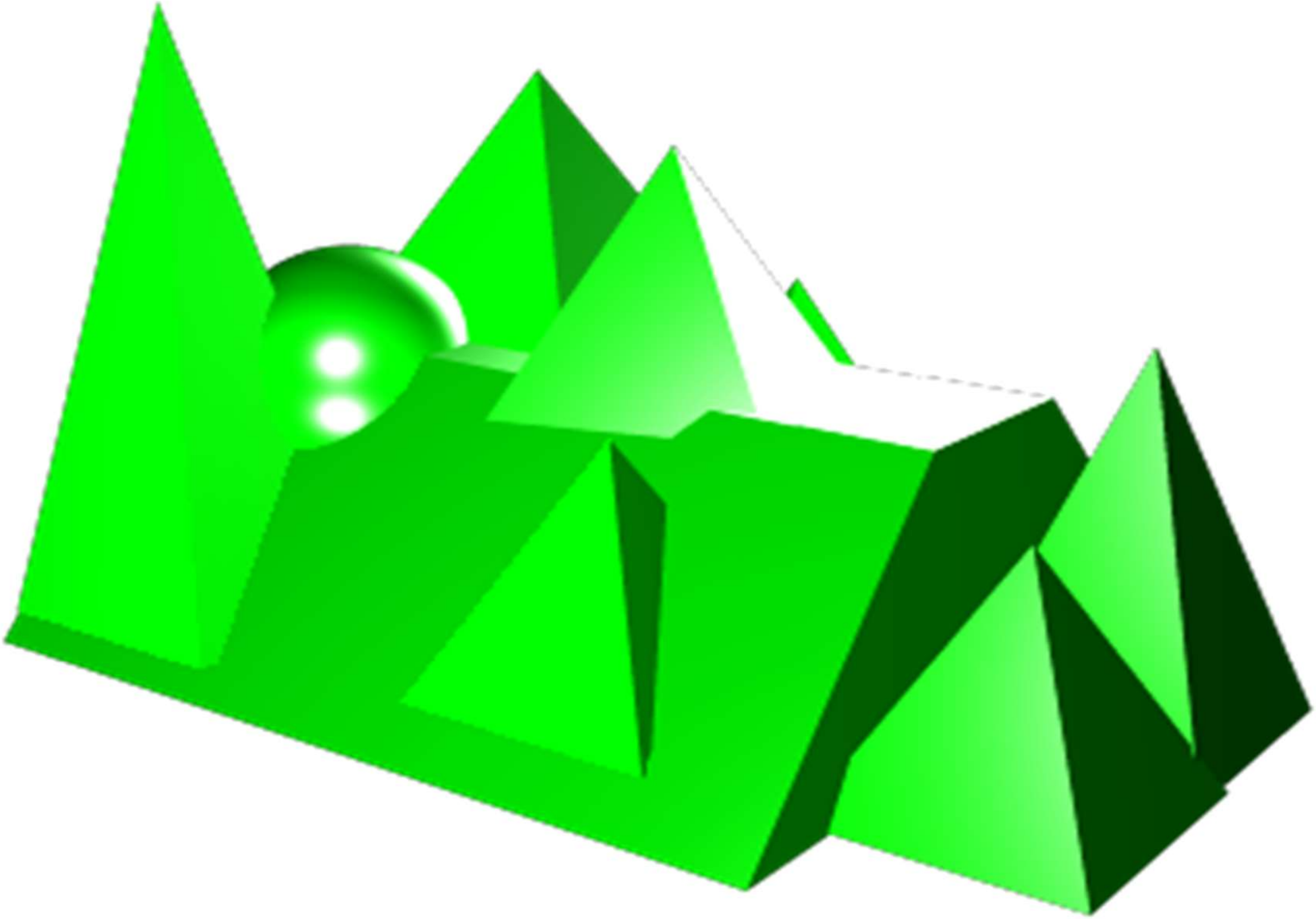
- A large variety of landscapes have been encountered
- The type of landscape cannot be known a priori
- Features which have been observed include
  - Multi-dimensionality
  - Plateaux
  - Discontinuities
  - Definition gaps
  - Multi-modality
  - Noise

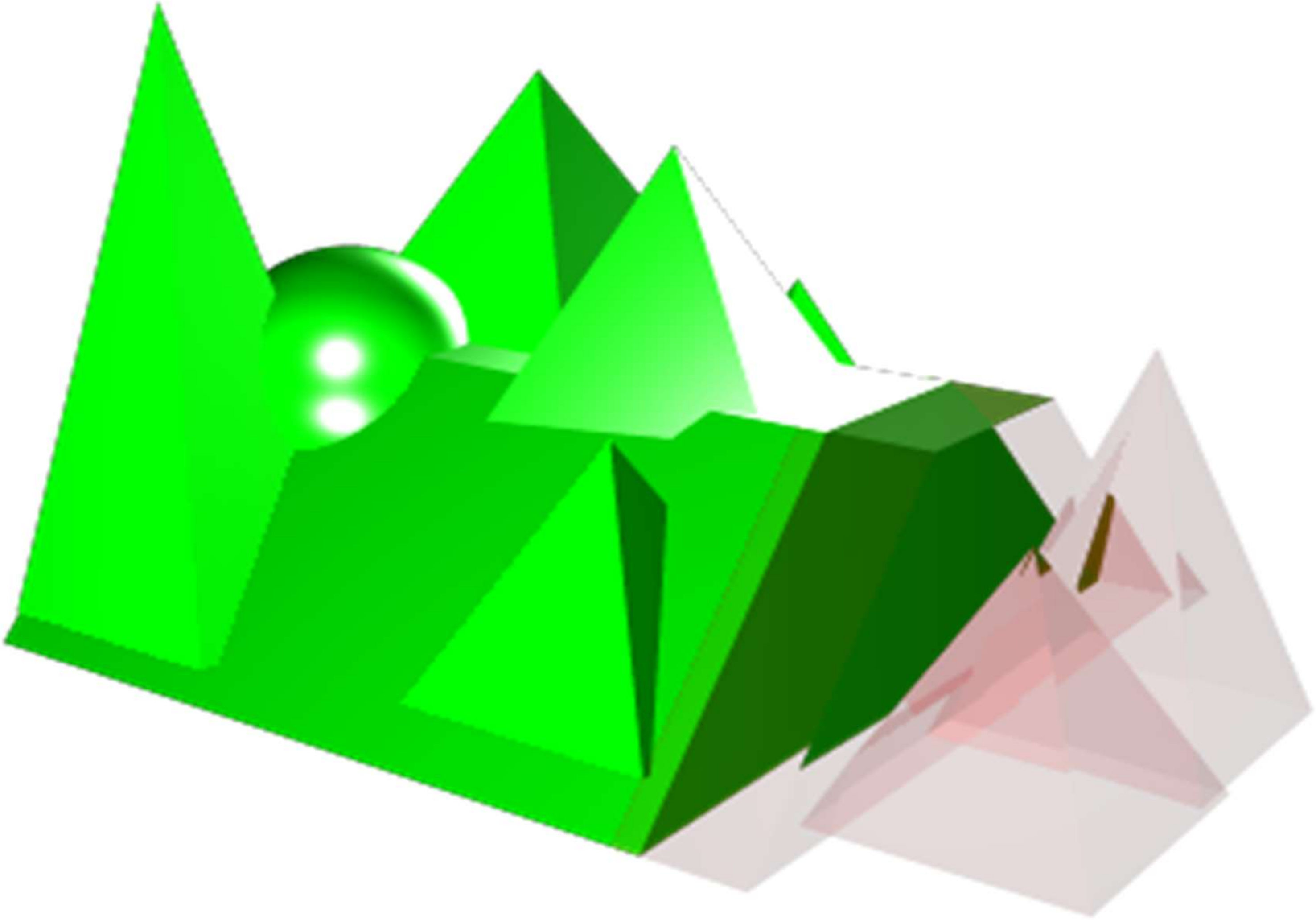
Managing Complexity
Advanced Search Techniques

- EA instead of GA
- multi-population model
- different search strategies (mutation, recombination)
- combination by migration
- more ressources for successful strategies

Managing Complexity
Search Space Size Reduction

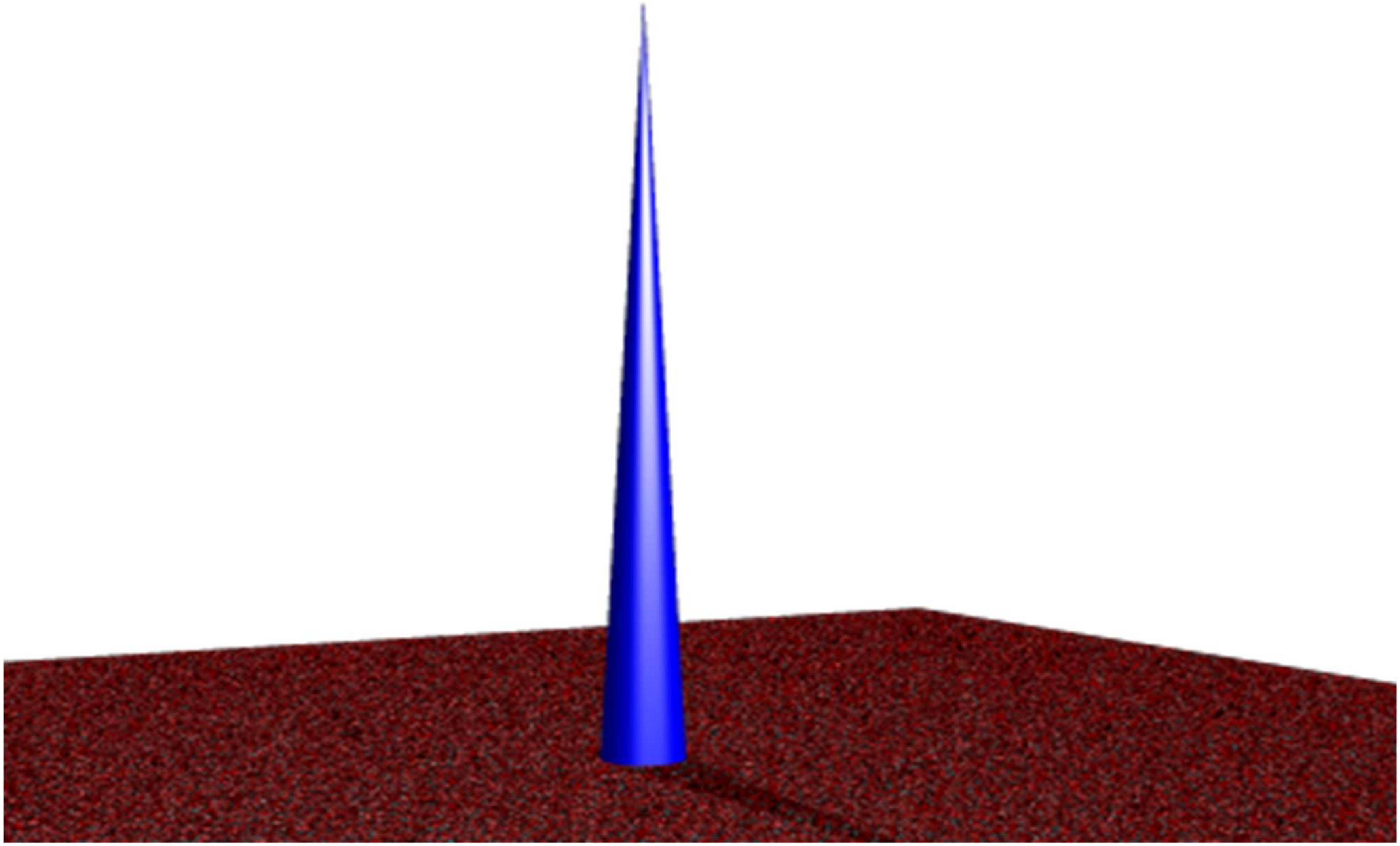
- The input to the program is our individual in a population
- However, the entire input may not affect the test goal
- There may be some input elements which cannot affect the test goal
- Static analysis can be used to find some of these redundant inputs
- This reduces the size of the search space

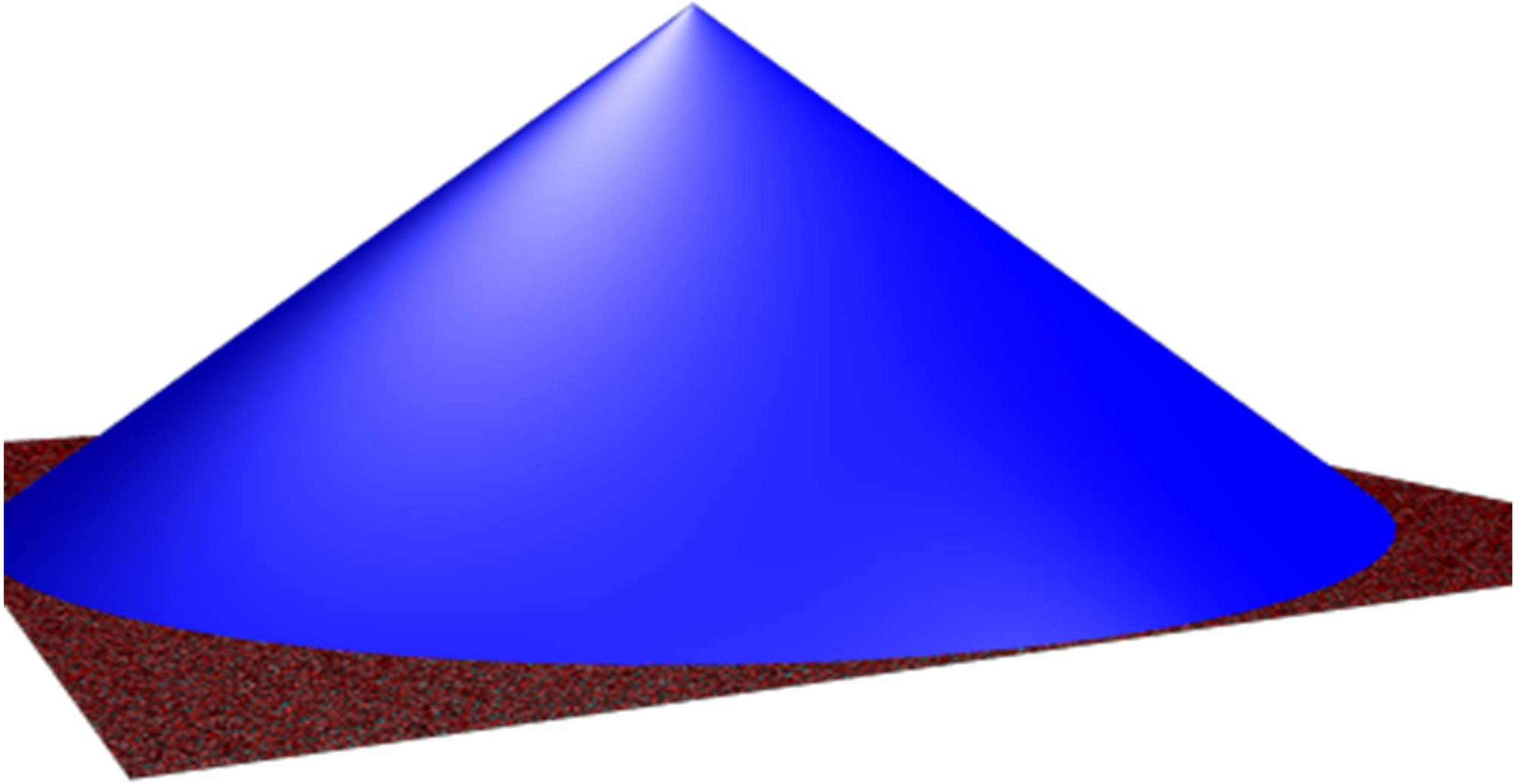




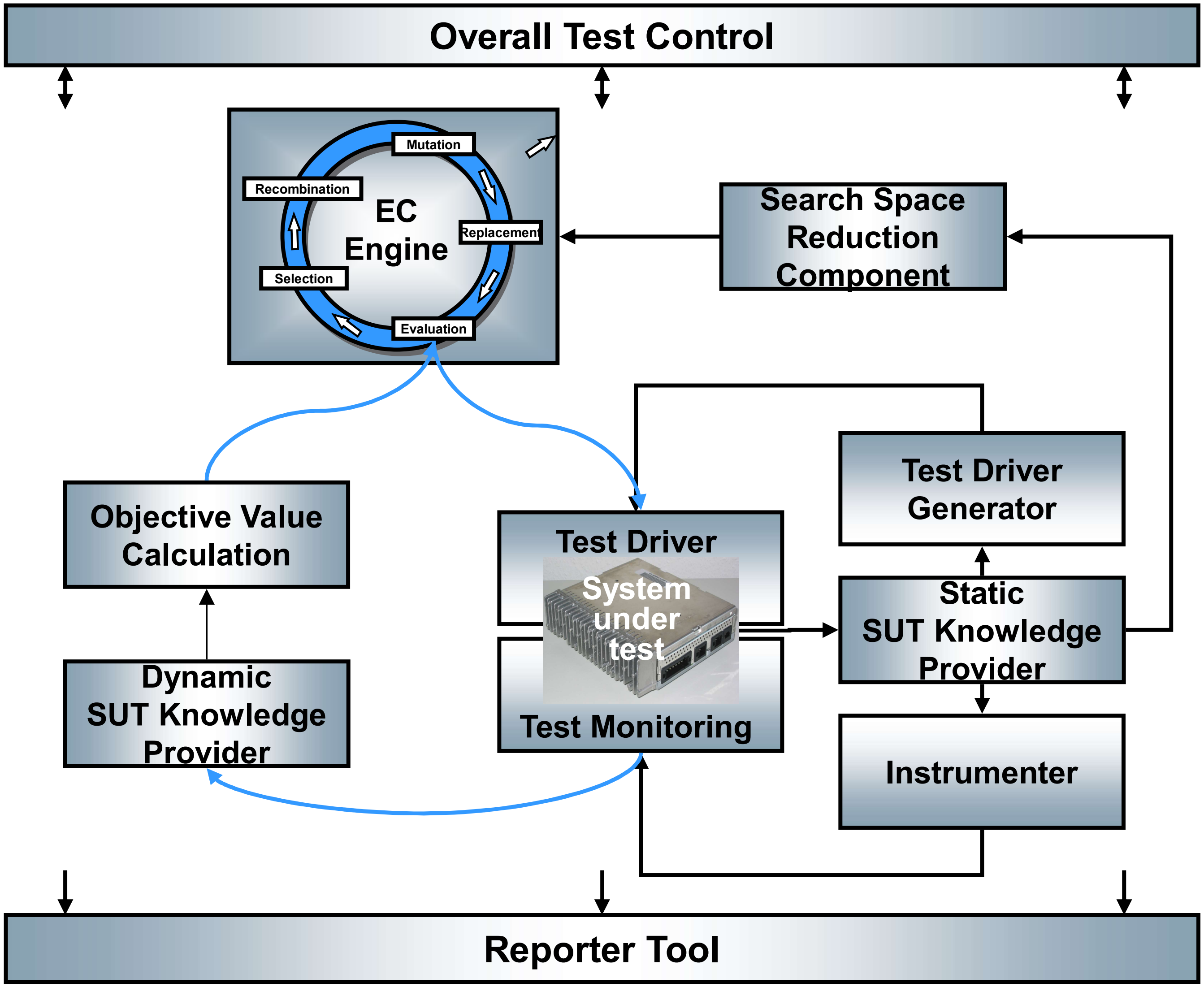
Managing Complexity
Search Space Transformation

- Some program features make search hard
- In the search community we would transform the landscape by transforming the fitness function
- Since our fitness function is applied to programs, we can transform the program in order to transform the fitness function
- One example is a flag which is either true or false. This creates a spike for structural testing
- The effect of the flag removal transformation on the landscape looks like this:





DC Evolutionary Testing Environment



Challenges

- Landscape transformation and simplification
- Integration of evolutionary testing and other analysis techniques (dynamic and static analyses)
- (Automatic) selection and configuration of search technique
- Testing complex systems with internal states and continuous behaviour
- Avoid the generation of invalid test data (generate new individual and replace, map to valid test datum, execute as robustness test)
- Definition of suitable stopping criteria for the test (convergence of population)
- Determining the quality of the generated tests (coverage)
- Gaining knowledge about the system under test from the large set of executed test data, observed landscapes, and search behaviour
- Determining the quality of the system under test

Conclusions

- Testing is very important to software quality assurance
- Without automation it is very expensive

- Testing provides a rich field of applications for Metaheuristics

- Many interesting and challenging issues arise
  - representation and transformation of individuals into test scenarios
  - fitness function definitions
  - every kind of landscape

Conclusions

- Testing is very important to software quality assurance
- Without automation it is very expensive

- Testing provides a rich field of applications for Metaheuristics

- Many interesting and challenging issues arise
  - representation and transformation of individuals into test scenarios
  - fitness function definitions
  - every kind of landscape

COR Special Issue on SBSE

- Search Based Testing is an example of Search Based Software Engineering
- There will be a COR focussed issue on Search Based Software Engineering
- Guest Editors: Walter Gutjahr and Mark Harman
- Deadline: 30th September
- *We may extend this by one or two weeks if sufficient requests*
- SBSE session next in room 32
- SBSE session after lunch (also room 32)